Machine Learning for Everyone.

Avneesh Jain (CodeKraft) and
Sambuddha Roy (LinkedIn)

What is Machine Learning?

What is Machine Learning?

- Enable computers/machines to “learn” from existing (i.e. historical) data.
- What is the learning used for?
  - Predict - new data from old (for eg. classification)
  - Extract hidden structure (for eg. clustering)
  - Summarize data
  - ... many other use-cases

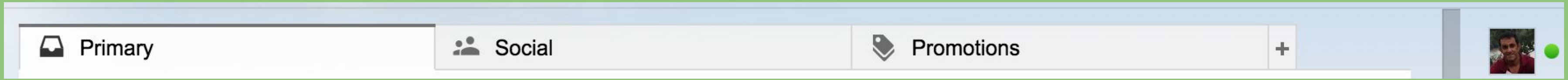
Machine Learning: Classification

According to wikipedia, "*classification* is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known."

Machine Learning: Classification

Examples abound:

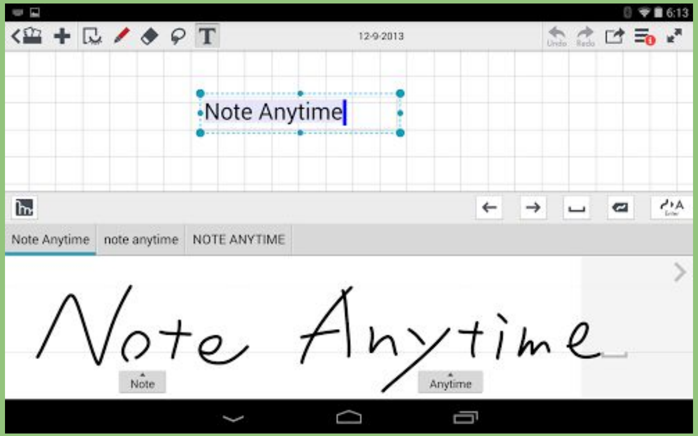
- Spam Classification
- GMail classifies your email as “Primary”, “Social” or “Promotions”



- “classification” of a newly bookmarked URL into the correct Bookmarks folder (was introduced as a feature for some time in Chrome)!

Machine Learning: Classification

- Handwriting Recognition.
- Speech Recognition



** Courtesy Google Images.

Machine Learning: Classification

- Cat or Dog?
- Car or Truck?

** Courtesy Google Images.

And Many other applications...

- Cancer diagnosis
- Video classification
- Click Stream Analysis

** Courtesy Google Images.

Many, Many…

- Internet Traffic Interception,
- Sentiment Analysis,
- …

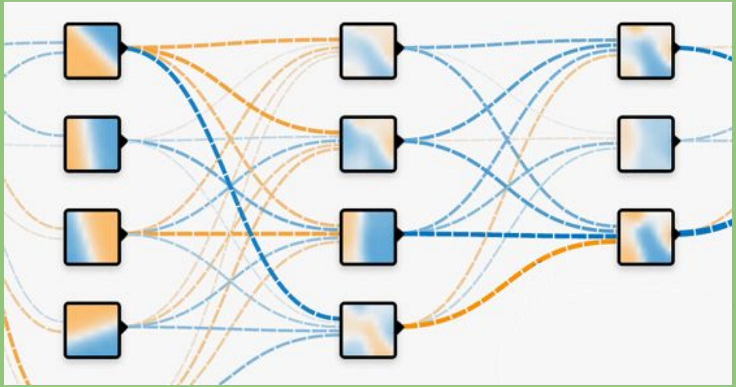
** Courtesy Google Images.

Roadmap i.e. topics we <u>will</u> cover

- Preliminaries for solving classification problems.
    - For this, we discuss features, decision boundaries,
    - Linear vs. non-linear classifiers.
- Discuss feature transformations
    - When a classifier is really linear, in some transformed features.
- Popular linear classifiers used in practice.
    - We discuss Naive Bayes in some detail.
    - Some details of logistic regression.
- Essential considerations:
    - Training, testing.
    - Metrics: AUC, Precision, Recall, F1-score, ...

Roadmap i.e. topics we <u>will</u> cover.

- Sentiment Analysis
  - Explain problem.
  - Describe dataset, separate into training and test datasets.
  - Train a Naive Bayes model to the problem.
- On to non-linear territory… Deep Learning.
  - Why is deep learning magical?



** Courtesy Google Images.

And Topics that we won't...

- Differences between
  - Supervised
  - Semi-supervised
  - Unsupervised
- Differences between
  - Regression
  - Classification
- Differences between
  - Discriminative
  - Generative
- Overfitting, regularization

And Topics that we won't...

- Bias-variance tradeoffs.
- Statistical Significance of parameters/weights
    - p-values etc.
- Cross-validation
    - Hold-out sets, etc.
- Correlated features.

And Topics that we won't...

- About deep networks:
    - Autoencoders, RBMs
    - RNNs, CNNs
    - GANs
    - Activation functions,
    - Dropout, etc.

Classification: how do we start?

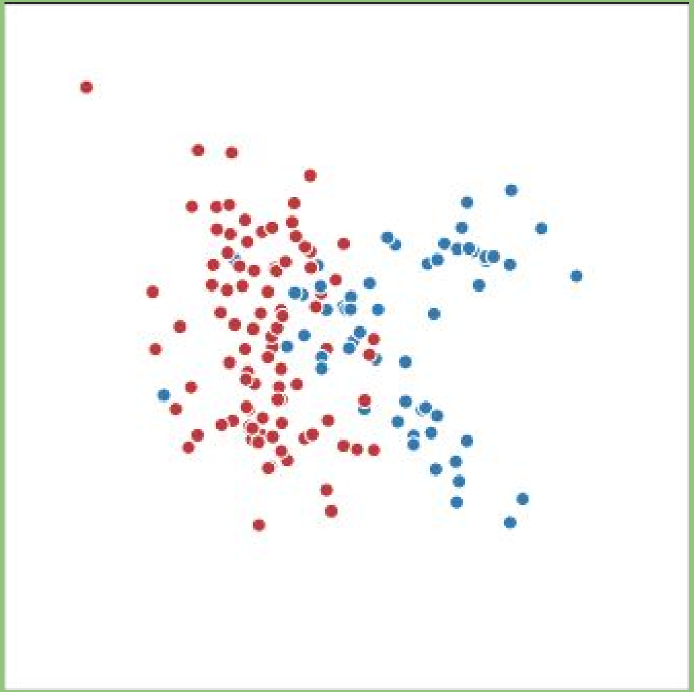
- Collect Features/Attributes!
- Eg. for the car vs. truck problem:
  - Number of wheels
  - Height of the vehicle
  - Length of the vehicle
  - Radius of wheels
  - Thickness of the wheels
  - What else?

Classification: how do we start?

- We can thereby “represent” an object (for instance a Buick) in “feature-space”.
- For instance, a Buick = (4, 1.5m, 3m, 0.5m, 0.2m…)

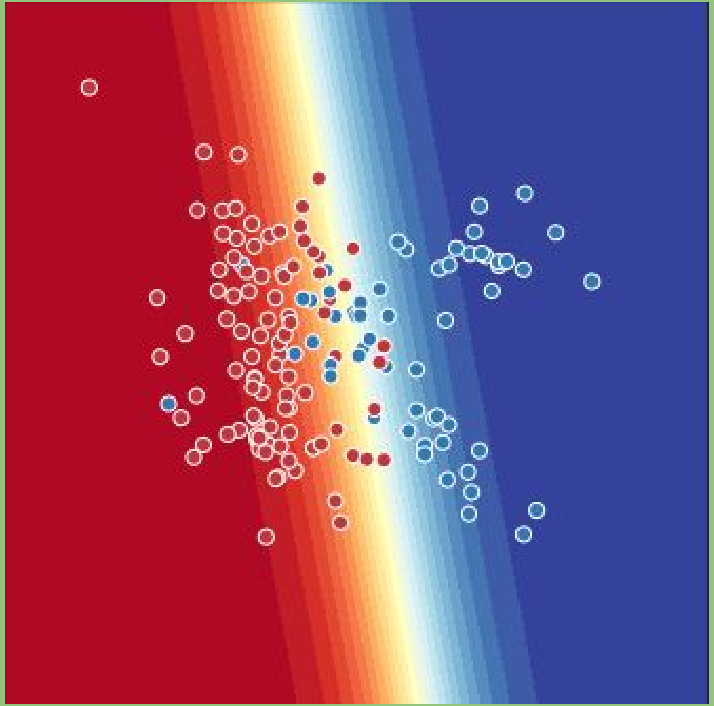
Feature space

Visualize the data: We plot the data points (according to their feature vectors) in n-diml. space.



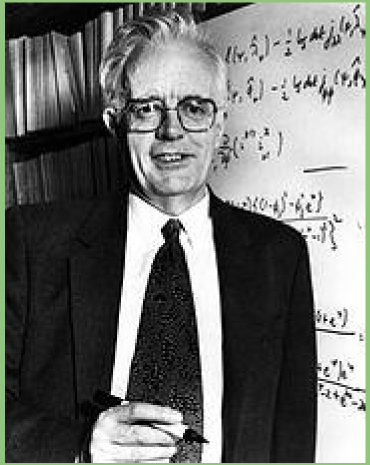
A linear classifier?

A linear classifier is one whose decision boundary is a line (or a hyperplane in feature space).



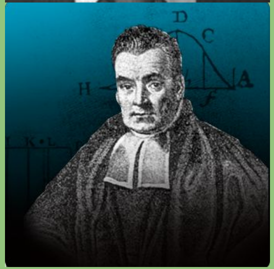
Examples of Linear Classifiers

- Logistic Regression
  - Most popular, used widely in diverse areas such as
    - *Advertising,*
    - *Fintech,*
    - *Many others.*



Examples of Linear Classifiers

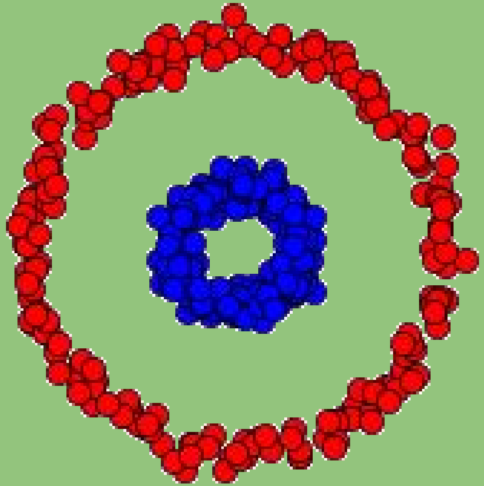
- Support Vector Machines
  - Sound theory backing, but heavier on optimization.
    - Image classification
    - Bio-informatics
- Naive Bayes.
  - Bayes’ed on Bayes’ Theorem. Used often in
    - Sentiment Analysis etc.



Do linear classifiers always suffice?

NO!

- What if the two classes are *not* separable by a hyperplane?



Do linear classifiers always suffice?

NO!

- What if the two classes are *not* separable by a hyperplane?
- Some problems allow ingenious constructions by which we can escape non-linearity. For instance, if it turns out that by a transformation:

Do linear classifiers always suffice?

NO!

- What if the two classes are *not* separable by a hyperplane?
- Some problems allow ingenious constructions by which we can escape non-linearity. For instance, if it turns out that by a transformation:
    - (x1, x2) -> (log x1, log x2), the classes become *linearly separable*!

Do linear classifiers always suffice?

NO!

- What if the two classes are *not* separable by a hyperplane?
- Some problems allow ingenious constructions by which we can escape non-linearity. For instance, if it turns out that by a transformation:
  - (x1, x2) -> (log x1, log x2), the classes become *linearly separable*!
- How do we figure out the right transformation?

Do linear classifiers always suffice?

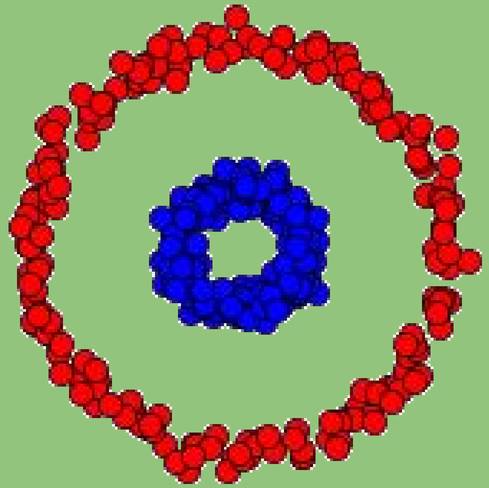
NO!

- What if the two classes are *not* separable by a hyperplane?
- Some problems allow ingenious constructions by which we can escape non-linearity. For instance, if it turns out that by a transformation:
  - (x1, x2) -> (log x1, log x2), the classes become *linearly separable*!
- How do we figure out the right transformation?
  - Intelligent feature engineering
  - Rigorous experimentation, accompanied with evaluation of metrics, etc.
  - Smart guesswork

Do linear classifiers always suffice?

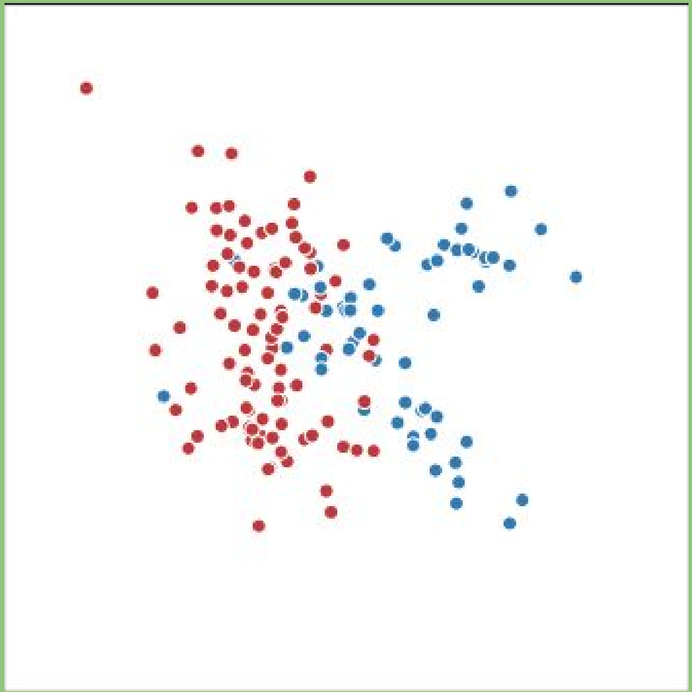
NO!

- What if the two classes are *not* separable by a hyperplane?
- We will come back to this in a moment.



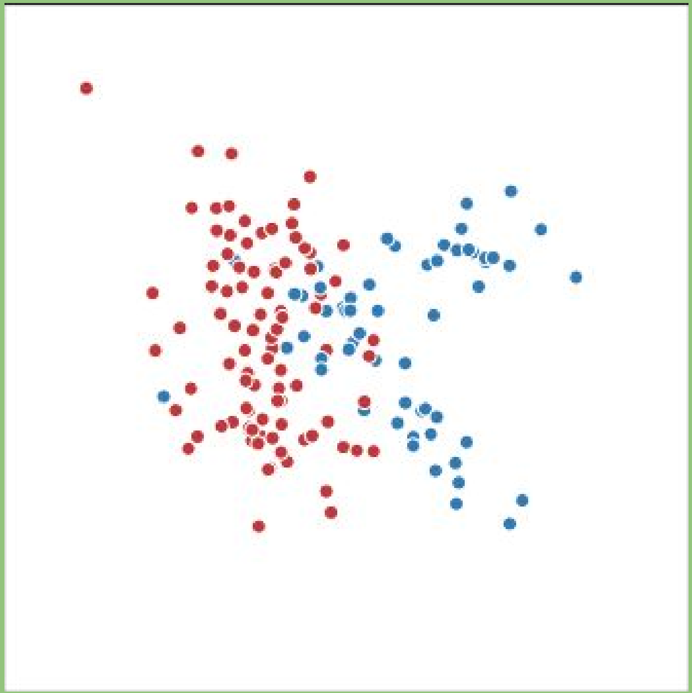
Back to Linear Classifiers

- What we we trying to learn?
  - The parameters determining the line.



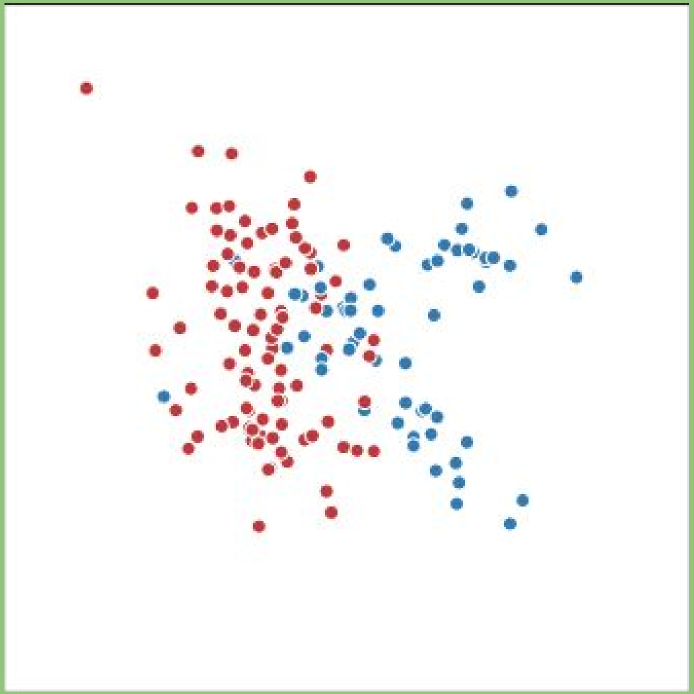
Back to Linear Classifiers

- What we we trying to learn?
  - The parameters determining the line.
- Which line? How do we make the choice?
  - Associate an objective function.



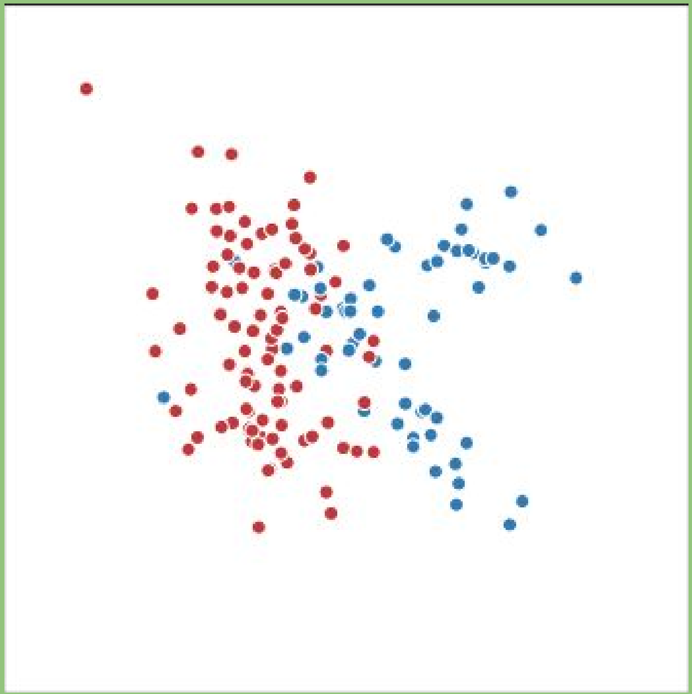
Back to Linear Classifiers

- What we we trying to learn?
  - The parameters determining the line.
- Which line? How do we make the choice?
  - Associate an objective function.
- However we are allowed to...
  - Transform features!



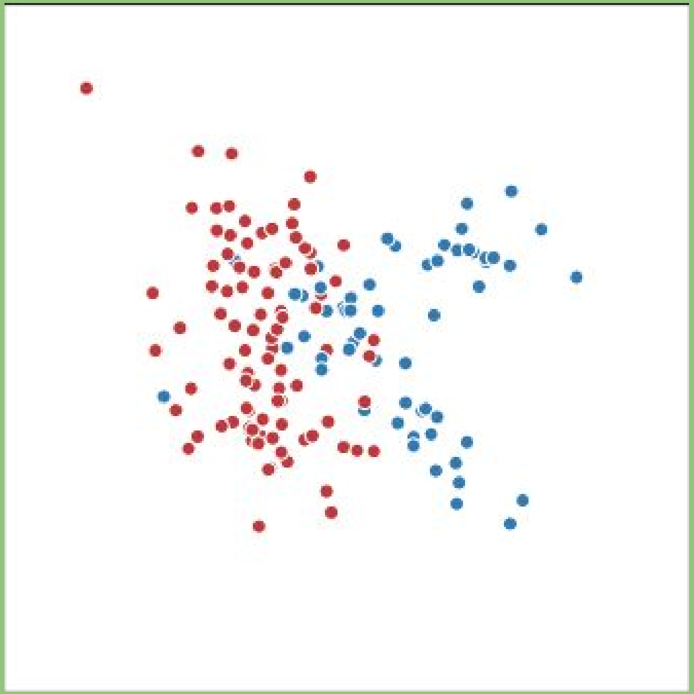
Feature Transformation.

- Suppose we were to "transform" features



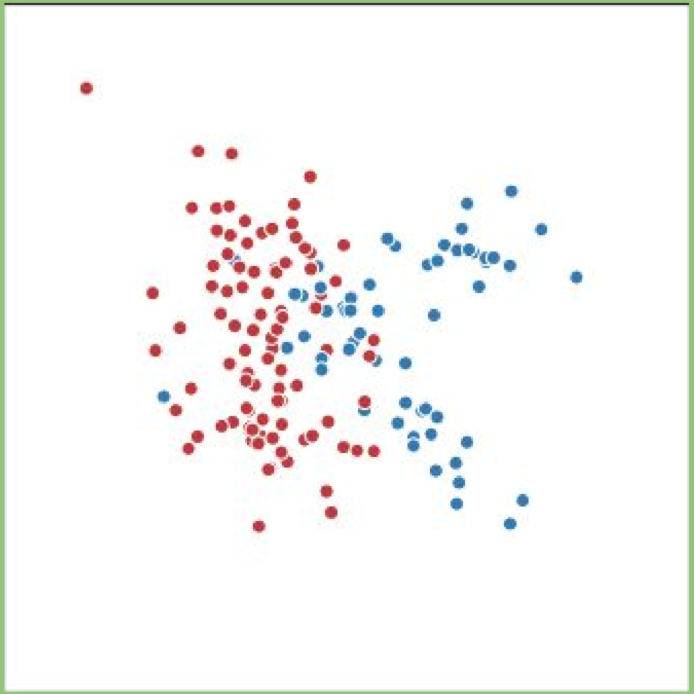
Feature Transformation.

- Suppose we were to “transform” features x, y
  - x -> 2 x (i.e. x_new = 2 x_old)
  - x -> (x + y)
  - I.e. linear transforms
  - Corresponds to “stretches” of featurespace
  - Linear stays linear.



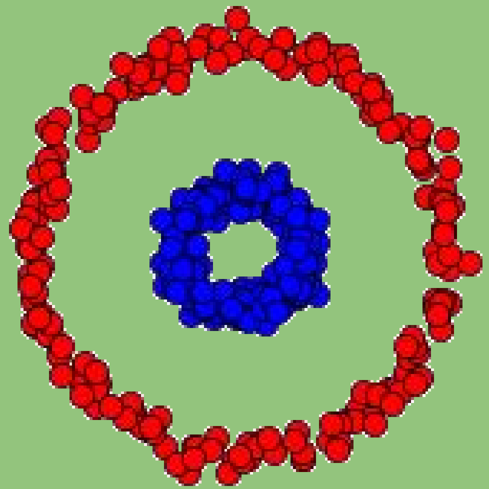
Feature Transformation.

- Hmm… still linear.



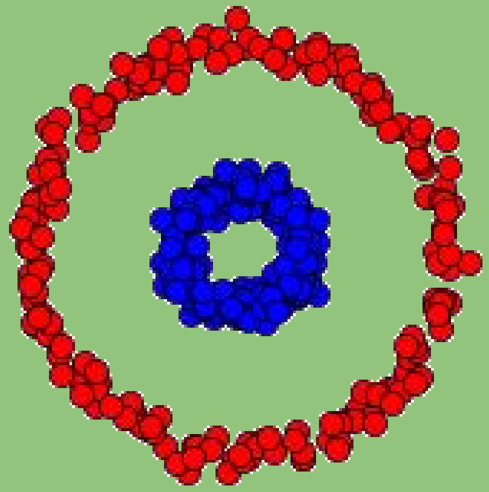
Feature Transformation.

- How about...



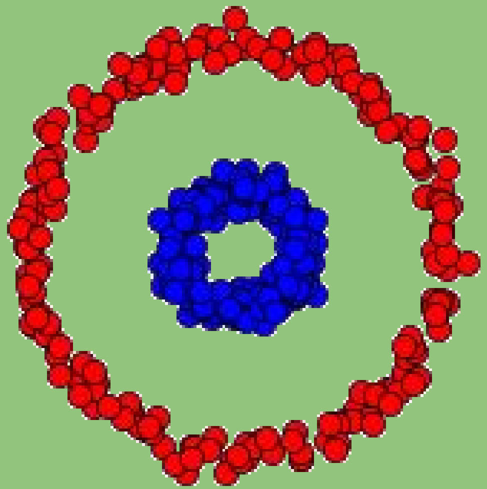
Feature Transformation.

- How about…
- How do we transform the (x, y) features here?



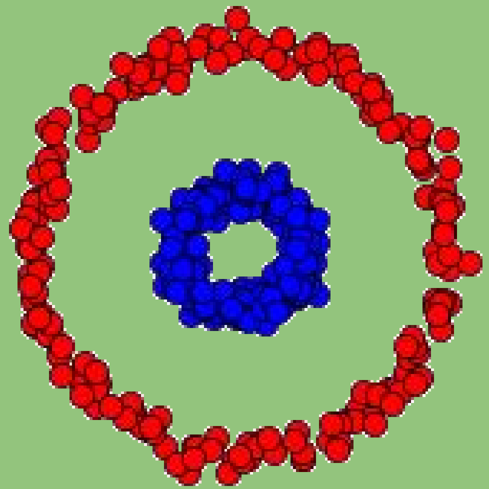
Feature Transformation.

- How about...
- How do we transform the (x, y) features here?
  - `x -> x^2`
  - `y -> y^2`



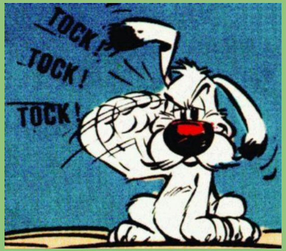
Feature Transformation.

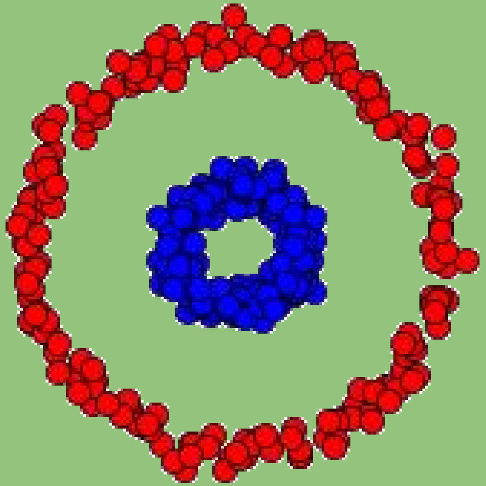
- How about...
- How do we transform the (x, y) features here?
  - x -> x^2
  - y -> y^2
- Uh, but this is *drawkcab*!
  - We visualized the data-set
  - Then figured out the right feature transformation!



Feature Transformation.

- How about...
- How do we transform the (x, y) features here?
    - $x \rightarrow x^2$
    - y -> y^2
- Uh, but this is *drawkcab*!
    - We visualized the data-set
    - Then figured out the right feature transformation!





Take-away

- Linear classifiers are stronger than they seem.
- Feature engineering is necessary at times.

Take-away

- Linear classifiers are stronger than they seem.
- Feature engineering is necessary at times.

Hmm.. How do we evaluate classifiers though?

Evaluation of models

How do we evaluate a trained model?

- Precision
- Recall
- Accuracy
- ...

Evaluation of models

- The confusion matrix

		Predicted label	
		Y	N
Actual label	Y	34	7
	N	9	45

Evaluation of models

- The confusion matrix

		Predicted label	
		Y	**N**
Actual label	**Y**	(TP) 34	(FN) 7
	N	(FP) 9	(TN) 45

- Accuracy = (TP + TN)/ALL
  - = 79/95 = 83%

Evaluation of models

- The confusion matrix

		Predicted label	
		Y	**N**
Actual label	**Y**	(TP) 34	(FN) 7
	N	(FP) 9	(TN) 45

- Precision = TP/(TP + FP)
  - = 34/43 = 79%

Evaluation of models

- The confusion matrix

Predicted label

Actual label		Y	N
	Y	(TP) 34	(FN) 7
	N	(FP) 9	(TN) 45

- Recall = TP/(TP + FN)
  - = 34/41 = 82%

Evaluation of models

- The confusion matrix

		Predicted label	
		Y	**N**
Actual label	**Y**	(TP) 34	(FN) 7
	N	(FP) 9	(TN) 45

- Accuracy = (TP + TN)/ALL
  - = 79/95 = 83%
- Precision = TP/(TP + FP)
  - = 34/43 = 79%
- Recall = TP/(TP + FN)
  - = 34/41 = 82%

Evaluation of models: Riddle

- The confusion matrix

		Predicted label	
		Y	**N**
Actual label	**Y**	(TP) ?	(FN) ?
	N	(FP) ?	(TN) ?

- Accuracy = (TP + TN)/ALL
- Precision = TP/(TP + FP)
- Recall = TP/(TP + FN)

Design a confusion matrix so that

- Accuracy is <u>high</u>
- Precision is <u>low</u>
- Recall is <u>low</u>

Sidewalk 1: Hands-on Logistic Regression

- We will conjure up some random data
- Then run logistic regression on this

- Tools used: sklearn, numpy.
- $ ipython notebook

SideWalk 2: Sentiment Analysis

What is sentiment analysis?

- From wikipedia, *“to extract subjective information in source materials”*.
- Why is it useful?
  - Also called opinion mining.
  - Marketing Research
  - Customer feedback
  - Ratings for movies, hotels, books etc.

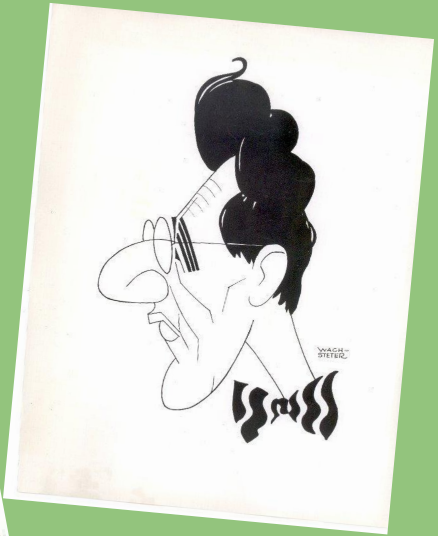
SideWalk 2: Sentiment Analysis

What is sentiment analysis?

- From wikipedia, *“to extract subjective information in source materials”*.
- Quiz - who said what?
  - “This is not a novel to be tossed aside lightly. It should be thrown with great force”
  - “ From the moment I picked your book up until I laid it down, I was convulsed with laughter. Someday I intend reading it.”
  - “When I was a kid, my parents moved a lot, but I always found them.”
  - “I watched this play at a disadvantage. The curtain was up.”

SideWalk 2: Sentiment Analysis

Sentimental people:



Task: to build a sentiment analyzer

We need:

- Labeled data!
  - Labels are “Positive”, “Negative”, and “Neutral”.
  - How much data?
- Partition labeled data into
  - Training
  - Test
- We will use IMDb movie reviews, and test it against new movies:
  - Dr. Strange
  - Arrival

Task: to build a sentiment analyzer

We need:

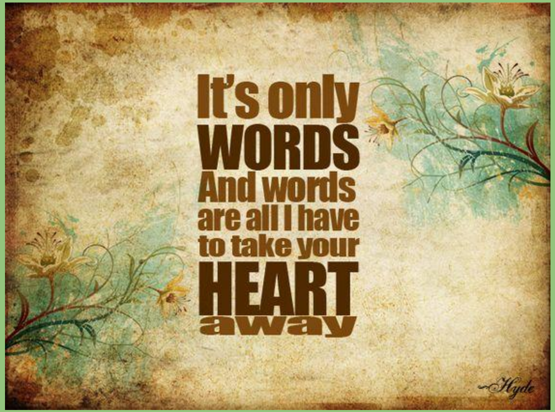
- Classifiers of choice:
  - Logistic Regression.
  - Naive Bayes. Based on Bayes' theorem.
- Metrics:
  - Precision, Recall, Accuracy.

Hand-over to Avneesh for AWS ML.

Task: to build a sentiment analyzer

We need:

- Features?
  - Words!
  - And combinations
    - Bigrams
    - Trigrams
    - Orthogonal sparse bigrams, etc.



What is... Bayes' theorem?

Bayes Theorem relates a conditional probability and its reverse.

Wait, wha....t?

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

First mind warp - why probabilities?!?

Most of the classifiers that we will see today are what are called

- Probabilistic classifiers

First mind warp - why probabilities?!?

Most of the classifiers that we will see today are what are called

- Probabilistic classifiers
- I.e. they do not just tell us if an item belongs to this class or that, but instead gives a score (between 0 and 1)!

First mind warp - why probabilities?!?

Most of the classifiers that we will see today are what are called

- Probabilistic classifiers
- I.e. they do not just tell us if an item belongs to this class or that, but instead gives a score (between 0 and 1)!
- Think of these as "probabilities" of belonging to one class or the other.

First mind warp - why probabilities?!?

Most of the classifiers that we will see today are what are called

- Probabilistic classifiers
- I.e. they do not just tell us if an item belongs to this class or that, but instead gives a score (between 0 and 1)!
- Think of these as “probabilities” of belonging to one class or the other.
- Overkill?

First mind warp - why probabilities?!?

Most of the classifiers that we will see today are what are called

- Probabilistic classifiers
- I.e. they do not just tell us if an item belongs to this class or that, but instead gives a score (between 0 and 1)!
- Think of these as "probabilities" of belonging to one class or the other.
- Overkill?
  - Not at all – simply good design.
  - Lots of times this is only an upstream scoring that goes into downstream decision making.

Idea behind Naive Bayes

Essential question for classification:

- Given a sentence “S” is it positive or negative (forget about neutral for now)?

Idea behind Naive Bayes

Essential question for classification:

- Given a sentence “S” is it positive or negative (forget about neutral for now)?
- I.e. in terms of probability, we are trying to score
    - P(“positive” | S) and also hence
    - P(“negative” | S)

Idea behind Naive Bayes

Essential question for classification:

- Given a sentence “S” is it positive or negative (forget about neutral for now)?
- I.e. in terms of probability, we are trying to score
    - P(“positive” | S) and also hence
    - P(“negative” | S)
- What does Bayes’ theorem do for us?

Idea behind Naive Bayes

Essential question for classification:

- Given a sentence "S" is it positive or negative (forget about neutral for now)?
- I.e. in terms of probability, we are trying to score
  - P("positive" | S) and also hence
  - P("negative" | S)
- What does Bayes' theorem do for us?
- Connects this up with P(S | "positive"), P(S | "negative")!

Idea behind Naive Bayes

Repeat in English

- Connects this up with P(S | “positive”), P(S | “negative”)!
- Consider all the “positive” sentences in the labeled data, how often do we expect to “find something like” S in that data?

Idea behind Naive Bayes

Repeat in English

- Connects this up with P(S | “positive”), P(S | “negative”)!
- Consider all the “positive” sentences in the labeled data, how often do we expect to “find something like” S in that data?
- What does “finding something like” S mean?

Idea behind Naive Bayes

Repeat in English

- Connects this up with P(S | “positive”), P(S | “negative”)!
- Consider all the “positive” sentences in the labeled data, how often do we expect to “find something like” S in that data?
- What does “finding something like” S mean?
    - If we are trying to look for something pretty close to S, in the labeled data, then chances are we will not find it. People are expressive, there are too many ways to describe something good (or bad).
    - Called the “sparsity problem”.
    - So we should look at the “ingredients” that go to make up S instead!

Idea behind Naive Bayes

Repeat in English

- Connects this up with P(S | “positive”), P(S | “negative”)!
- Consider all the “positive” sentences in the labeled data, how often do we expect to “find something like” S in that data?
- What does “finding something like” S mean?
    - Well, really, the words in S.
    - Say, if S = “This bike is good”. How often do we expect to see the word “good” among the positively labeled sentences in our training data?

Idea behind Naive Bayes

Other details:

- (conditional) independence assumption
  - The words in a document/sentence are “conditionally” independent given the class.
  - Example?
- This takes care of
  - The sparsity problem
  - Breaks up the computation of probabilities into manageable pieces.

Idea behind Naive Bayes

Other details:

- (conditional) independence assumption
  - The words in a document/sentence are “conditionally” independent given the class.
  - Example?
- This takes care of
  - The sparsity problem
  - Breaks up the computation of probabilities into manage-able pieces.
- However, new problem arises:
  - What if we haven’t seen a word at all? How do we manage that?

Idea behind Naive Bayes

Other details:

- However, new problem arises:
    - What if we haven't seen a word at all? How do we manage that?

Idea behind Naive Bayes

Other details:

- However, new problem arises:
  - What if we haven't seen a word at all? How do we manage that?
- We don't allow these conditional probabilities to vanish to zero.
  - Essentially jerky behavior:
    - If word present, then non-zero conditional probability
    - If word not seen as yet, jumps to zero conditional probability.
  - Apply *smoothing*.

Enough! Show me the code!

Enough! Show me the code!

Enough! Show me the code!

```
import re, math, collections, itertools, os
import nltk, nltk.classify.util, nltk.metrics
from nltk.classify import NaiveBayesClassifier
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
from nltk.metrics import precision
from nltk.metrics import recall
import random
import csv
```

Sidewalk 3: Deep Learning

Recall the issues with feature engineering.

- We needed to know the data well, to know the data well (i.e. to classify it).
- What if we were able to “learn” the (at times non-linear) features to be used?

Sidewalk 3: Deep Learning

Recall the issues with feature engineering.

- We needed to know the data well, to know the data well (i.e. to classify it).
- What if we were able to “learn” the (at times non-linear) features to be used?
- That is precisely what Deep Learning tries to do.

Sidewalk 3: Deep Learning

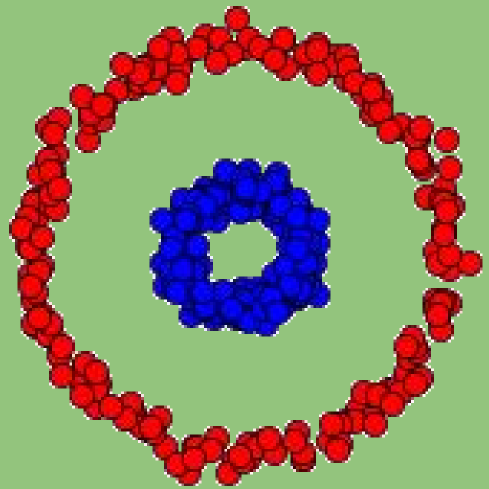
- That is precisely what Deep Learning tries to do.
- Stacks classifiers on top of classifiers.
- Outputs of earlier “coarser” classifiers are non-linear inputs to the latter layers.

Sidewalk 3: Deep Learning

- That is precisely what Deep Learning tries to do.
- Stacks classifiers on top of classifiers.
- Outputs of earlier “coarser” classifiers are non-linear inputs to the latter layers.
- Look at the earlier example again.

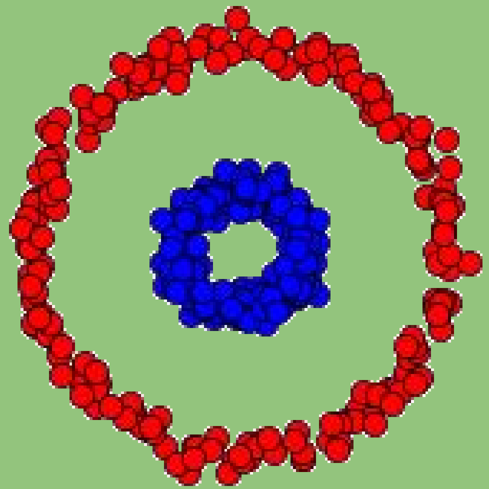
Sidewalk 3: Deep Learning

- That is precisely what Deep Learning tries to do.
- Stacks classifiers on top of classifiers.
- Outputs of earlier “coarser” classifiers are non-linear inputs to the latter layers.
- Look at the earlier example again.



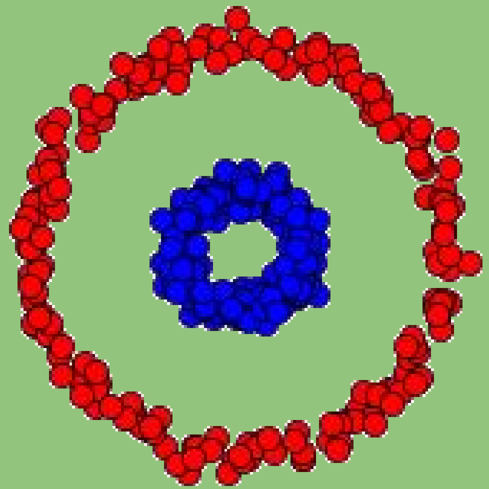
Sidewalk 3: Deep Learning

- Look at the earlier example again.
- Suppose we gave the machine features of the form
  - x^c without specifying c
- Would the machine be able to figure out for itself that c = 2?



Sidewalk 3: Deep Learning

- Look at the earlier example again.
- Suppose we gave the machine features of the form
  - x^c without specifying c
- Would the machine be able to figure out for itself that c = 2?



- Perhaps...
  - Given enough data.
  - Smarter training procedures.

Sidewalk 3: Deep Learning.

How does this tie up with the classifier-on-classifier story?

Well,

- a first level classifier is trying to figure out the “correct” feature transform x -> x^2, and
- the second level classifier is trying to run a line through the (thereby) modified data.

Sidewalk 3: Deep Learning.

How does this tie up with the classifier-on-classifier story?

Well,

- a first level classifier is trying to figure out the "correct" feature transform x -> x^2, and
- the second level classifier is trying to run a line through the (thereby) modified data.

Surprisingly simple, and stunningly powerful idea!

Well... all is hunkydory but...

- Note now that instead of the handcrafted x^2, we have a free parameter c in x^c.

Well... all is hunkydory but...

- Note now that instead of the handcrafted x^2, we have a free parameter c in x^c.
- A lot more parameters in a deep network!
  - Need to find these parameters – this is what "learning" amounts to.

Well… all is hunkydory but…

- A lot more parameters in a deep network!
    - Need to find these parameters – this is what “learning” amounts to.

Well... all is hunkydory but...

- A lot more parameters in a deep network!
  - Need to find these parameters – this is what “learning” amounts to.
- Insight from linear equations:
  - Underdetermined systems. Fewer equations/constraints than variables are a problem. Cannot have unique solutions.
  - Think of each training example as a “constraint” (constraining the network to output something close to the “label” on that training input).

Well... all is hunkydory but...

- A lot more parameters in a deep network!
  - Need to find these parameters – this is what "learning" amounts to.
- Insight from linear equations:
  - Underdetermined systems. Fewer equations/constraints than variables are a problem. Cannot have unique solutions.
  - Think of each training example as a "constraint" (constraining the network to output something close to the "label" on that training input).
- More the number of parameters, more the number of training/labeled examples required.
  - In the age of big data, plenty of unlabeled data.

Shallow vs. deep learning

- A shallow classifier (eg. SVM, Log Reg) is trying to learn a function
  - $y = F(x)$

- OTOH, a deep classifier (say of depth 3, 2 hidden layers) is trying to learn a function
  - $y = F \circ G \circ H(x)$, where "o" = composition.

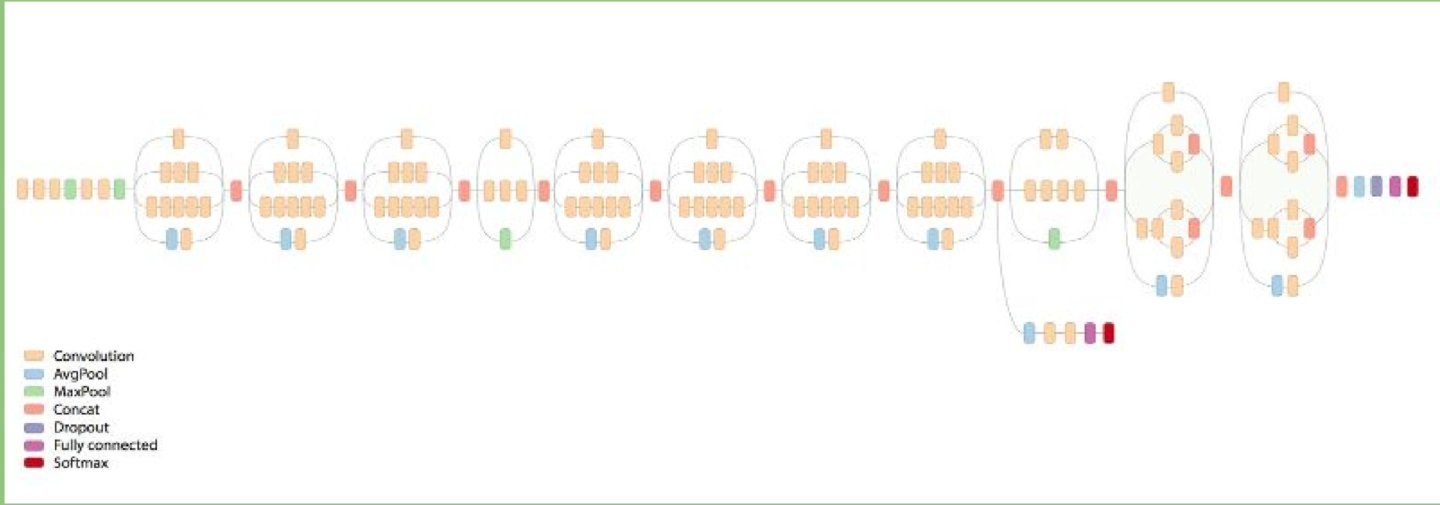
Shallow vs. deep learning - what gives?

- But F o G o H is just yet another function!
- ???

CNN - And now the news...

Deep Learning models for images.

- Recent fascinating work has used deep networks (involving what are called "convolutions") to classify images.
- Used a large publicly available training data set, called ImageNet.
- Error has dropped to ~3%!



Deep Learning models for Language.

- Recent fascinating work has resurrected recurrent neural nets, LSTMs for
  - Machine translation
  - Sentiment analysis
  - Language modeling
  - Many many others.

EndGame

- Please fill up survey with comments.
- We will feed it to this very sentiment analyzer and come up with an aggregate score!

EndGame

- Please fill up survey with comments.
- We will feed it to this very sentiment analyzer and come up with an aggregate score!

Just Kidding!

Thank You!!!

EndGame

Thank You!!!

Evaluation of models

- The confusion matrix

Actual label \ Predicted label	Y	N
Y	34	7
N	9	45